DOCKET No.

NAI1P067/01.266.01

# U.S. PATENT APPLICATION

# FOR

# USER-CONFIGURABLE NETWORK ANALYSIS DIGEST SYSTEM AND METHOD



ASSIGNEE: NETWORKS ASSOCIATES TECHNOLOGY, INC.

SILICON VALLEY IP GROUP

P.O. BOX 721120

SAN JOSE, CA 95172

# USER-CONFIGURABLE NETWORK ANALYSIS DIGEST SYSTEM AND METHOD

## FIELD OF THE INVENTION

The present invention relates to network analysis, and more particularly to reporting on network analysis.

## BACKGROUND OF THE INVENTION

Numerous tools have been developed to aid in network management involving capacity planning, fault management, network monitoring, and performance measurement. One example of such tools is the network analyzer.

In general, a "network analyzer" is a program that monitors and analyzes network traffic, detecting bottlenecks and problems. Using this information, a network manager can keep traffic flowing efficiently. A network analyzer may also be used to capture data being transmitted on a network. The term "network analyzer" may further be used to describe a program that analyzes data other than network traffic. For example, a database can be analyzed for certain kinds of duplication. One example of a network analyzer is the SNIFFER® analyzer device manufactured by NETWORK ASSOCIATES, INC®.

Network analyzers are often capable of collecting a vast array of network traffic information which may be monitored and analyzed. Unfortunately, the more complex a network scenario is, the sooner a network manager gets overloaded by all of the

collected information. Trying to deal with such large amount of information while attempting to focus on the right issues can be quite troublesome.

There is thus a need for a tool to make the process of receiving, organizing and analyzing network traffic information easier, faster, and more focused on a user's needs.

## DISCLOSURE OF THE INVENTION

A system, method and computer program product are provided for user-configured network analysis reporting. Initially, pluralities of templates are identified which are provided based on user input. Then, a database is queried for network traffic information based on the identified templates. The templates are then populated with the network traffic information and stored on disc (i.e. WEB Server directories, etc.). Next, the network traffic information is available for viewing over a network utilizing the final documents, which are a result of the populated templates.

Two different processes may be available to create customized reports for reporting purposes. Both use similar code, but work on different types of templates (i.e. first and second type). A first process is called "WARE," which stands for Web Automated Reporting Engine. WARE reads parameter files. It opens web page templates, queries for specific information from the database to populate the web page template with the required information. A new document is thus created with a plurality of placeholders replaced with real data. Such document is stored at a parameter-specific location in memory.

A second process is called CARE, which stands for CrystalTM Automated Reporting Engine. CARE also reads parameter files. It has an application program interface (API) for interfacing a conventional CrystalTM system. In use, a CrystalTM report template is opened, and CARE takes the information of the parameter file to feed into the CrystalTM system. Requested reports are created and automatically stored at a parameter-specific location in memory.

A graphical user interface (GUI) is provided to easily create and edit the parameter files, which may control the CARE and WARE processes. Both processes, CARE and WARE, can either run by a scheduler or manually by user control.

A flexible system is thus afforded to provide users from a network manager level up to the chief information officer (CIO) with the exact type of reports they need to control and run their network business. Every day they can receive their reports with the same layout, but current information.

The information may be stored in static files. This offers a very easy way to save and share reports by simply uploading complete directories on a CD or DVD. These can later be viewed completely independent from the creation system.

The network manager, for whom these customized reports are created, may see every day the exact and preferred layout of his or her report.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure **1** illustrates a network architecture, in accordance with one embodiment.

Figure **2** shows a representative hardware environment that may be associated with the data servers and computers of Figure **1**, in accordance with one embodiment.

Figure **3** illustrates an exemplary architecture for user-configurable reporting, in accordance with one embodiment.

Figure **4** illustrates a method for user-configurable network analysis reporting, in accordance with one embodiment.

Figure **5** illustrates a method for generating a user configurable report with a Web Automated Reporting Engine (WARE), in accordance with operation **422** of Figure **4**.

Figure **5A** illustrates an exemplary template to be used for WARE input.

Figure **5B** illustrates an exemplary output file, created by WARE.

Figure **6** illustrates a method for generating a user configurable report with a CrystalTM Automated Reporting Engine (CARE), in accordance with operation **422** of Figure **4**.

Figure **7** illustrates an exemplary graphical user interface capable of displaying the populated templates over a network.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure **1** illustrates a network architecture **100**, in accordance with one embodiment. As shown, a plurality of networks **102** is provided. In the context of the present network architecture **100**, the networks **102** may each take any form including, but not limited to a local area network (LAN), a wide area network (WAN) such as the Internet, etc.

Coupled to the networks **102** are data server computers **104** which are capable of communicating over the networks **102**. Also coupled to the networks **102** and the data server computers **104** is a plurality of end user computers **106**. In the context of the present description, a computer may refer to any web server, desktop computer, lap-top computer, hand-held computer, printer or any other type of hardware/software.

In order to facilitate communication among the networks **102**, at least one gateway **108** is coupled therebetween. It should be noted that each of the foregoing network devices as well as any other unillustrated devices may be interconnected by way of a plurality of network segments. In the context of the present description, a network segment includes any portion of any particular network capable of connecting different portions and/or components of a network.

Coupled to any one of the foregoing components and/or segments may be a network analyzer. One exemplary network analyzer that may be used is the SNIFFER® analyzer device manufactured by NETWORK ASSOCIATES, INC®. In use, the network analyzer is adapted for monitoring and analyzing network traffic, detecting bottlenecks and problems. In addition to or instead of such functionality, the network analyzer may be adapted for user-configurable network analysis reporting.

Initially, a plurality of templates are identified which are provided based on user input. This may be accomplished by creating new templates or selecting pre-existing templates before or at the time of reporting.

Then, a database is queried for network traffic information based on the identified templates. The templates are then populated with the network traffic information and stored on WEB server directories. Next, the network traffic information is reported over a network utilizing the populated templates.

By this design, a user may focus the reporting and analysis of collected network traffic information in way that removes superfluous and/or irrelevant information. More information relating to an exemplary application of the foregoing principles will be set forth hereinafter in greater detail.

Figure **2** shows a representative hardware environment that may be associated with the data server computers **104** and/or end user computers **106** of Figure **1**, in accordance with one embodiment. Such figure illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit **210**, such as a microprocessor, and a number of other units interconnected via a system bus **212**.

The workstation shown in Figure **2** includes a Random Access Memory (RAM) **214**, Read Only Memory (ROM) **216**, an I/O adapter **218** for connecting peripheral devices such as disk storage units **220** to the bus **212**, a user interface adapter **222** for connecting a keyboard **224**, a mouse **226**, a speaker **228**, a microphone **232**, and/or other user interface devices such as a touch screen (not shown) to the bus **212**, communication adapter **234** for connecting the workstation to a communication network **235** (e.g., a

data processing network) and a display adapter **236** for connecting the bus **212** to a display device **238**.

The workstation may have resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. It will be appreciated that a preferred embodiment may also be implemented on platforms and operating systems other than those mentioned. A preferred embodiment may be written using JAVA, C, and/or C++ language, or other programming languages, along with an object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications.

Figure **3** illustrates an exemplary architecture. **300**, which can be implemented on a Network Associates® Sniffer® reporting server, presents an overall reporting system and graphical user interface (GUI), in accordance with one embodiment. All general functions may be available (but not exclusive) through this interface.

A database **302** may be provided which is consistently populated with network traffic information. In the context of the present description, such network traffic information may include any information relating to communications over a network. The population of the database **302** may be accomplished in any desired manner. For example, a plurality of remote monitoring (RMON) agents may be used to create "csv" files (i.e. comma separated value files). In use, the agents continuously write data into csv files. These csv files may then be used to populate the database **302**. Typically, in every time interval, which can be minutes, hours, or days; new statistical data may be imported in the database **302**.

It should be noted that the database **302** includes a plethora of information including first portions that may be relevant to a first user, second portions that may be relevant to a second user, and so forth. As will soon become apparent, the present architecture **300** allows a user to configure the manner in which the network traffic information is extracted from the database **302** and reported.

A Web Automated Reporting Engine (WARE) process **312** works on parameter files **311** which can look like the one shown in Table #1. A detailed description will follow later.

Table #1

```
[DATABASE] Access_User
[INPUT]..\reports\[DATE]\chapter7.htm
[EXPORT]..\reports\:YEAR:_:MONTH:_:DAY:\
[DATEFORMAT] %m/%d/%Y
[PARAMETERS]
:DATE: 08/07/2001
:SQL_Q1: Select segment from segments where time >=
CDate(':DATE: 00:00:00') and time <=CDate(':DATE: 23:59:59')
group by segment order by segment asc
:SQLOUT: REC_NR,SEGMENT
```

Based on all of this information, the WARE process **312** reads a template **314**, which may be a standard web file and includes several placeholders. The WARE process **312** may replace the placeholders with real dates and updated information, queried from the database **302**. A new updated file **320** is now stored in some directory, which name can also be built of variables like a date, a segment, or report type. A special Sniffer® Digest Homepage **324** can now refer to this predefined structure of the reports and their locations.

A CystalTM Automated Reporting Engine (CARE) process **316** works on parameter files **310**, which can look like the one shown in Table #2. A detailed description will follow later.

Table #2

```
[REPORT]d:\reports\crystaltemplates\report.rpt

[EXPORT]d:\reports\:YEAR:_:MONTH:_:DAY:\:SEGMENT:\:HOST:\default
.htm


[SELECT] {HOSTS.Host} = ":HOST:" and
         {HOSTS.Time} in DateTime (:YEAR:,:MONTH:,:DAY:,00, 00,
00) to DateTime (:YEAR:,:MONTH:,:DAY:,23, 59, 59) and
         {HOSTS.Segment} = ":SEGMENT:"

[DATEFORMAT] %Y-%m-%d

[PARAMETERS]
:DATE: 2001-08-1
:DATE: 2001-08-2
:DATE: 2001-08-3

:SEGMENT: Dallas
:SEGMENT: NewYork
:SEGMENT: NewYork-Atlanta
:SEGMENT: Dallas-Atlanta

:HOST: WEBServer1
:HOST: SAPServer1
```

Based on this information, the CARE process **316** connects with a programming interface to a CrystalTM library. It then forwards the CrystalTM report template, which may be a RPT file **318**, and several parameters to a CrystalTM system for report generation. This way, the report can be narrowed down to only show the required information. The parameters, which CARE transfers to the CrystalTM system, can either be static values from the parameter file **310**, or result from some query into the database **302**. The CrystalTM system saves the report as some web accessible files **320**, which can be, for example, HTML files. A special Sniffer® Digest Homepage **324** can now refer to this predefined structure of the reports and their locations.

In one embodiment, the templates may be selected and tailored by the user. Such templates may include any data structure capable of dictating the manner in which the network traffic information is extracted from the database **302,** and/or reported to the user. To accomplish this, an interface **304** is provided for allowing a user to select reporting parameters which, in turn, determine the appropriate template.

In the context of the present description, a parameter may include any aspect of network traffic information reporting that may be configured by a user. Just by way of example, the parameters may include which network segments should be reported (i.e. which tier of network segments, etc.), a format of the reporting (i.e. pie-chart, graphs, customizable templates, etc.) or the like. Further, the parameters may indicate where the network traffic information comes from, what type of network traffic information is used, to what location the network traffic information is written, etc. Still yet, the parameters may indicate the time and period at which various template fields are populated.

An active server page **306** may be used for delivering the parameters from the user to a particular predetermined server. Further, a COM component **308** may be used to validate the parameters provided by the user to make sure the data is correct in a parameter file which is stored for subsequent use. Table #1 and table #2 illustrate two exemplary files **310** and **311** indicating a plurality of exemplary parameters.

Figure **4** illustrates a method **400** for user-configurable network analysis reporting, in accordance with one embodiment. As an option, the present method **400** may be carried out in the context of the architecture **300** set forth in Figure **3**. Of course, the present method **400** may be carried out in any desired context.

As shown, an interface is displayed in operation **402**. Such interface may include a network analysis reporting interface homepage like that mentioned earlier, or any other desired interface. Using such interface, a user may indicate whether he or she desires to operate in an edit mode or a report mode. See decision **404**. This may be accomplished by simply selecting various icons, defaulting in the report mode unless a user action indicates otherwise, etc.

If the method **400** is to operate in an edit mode, input is then received from a user. See operation **406**. Such input may take the form of the various parameters mentioned hereinabove. Of course, the input may take any form.

Next, a parameter file is generated based on the input in operation **408**. The contents of the parameter file may be validated with, for example, a COM component similar to that set forth during reference to Figure **3**. Note operation **410**. In operation **412**, the parameter file is stored for immediate or subsequent use.

If, on the other hand, the method **400** is to operate in a report mode, a user may be identified in operation **418**. This may be accomplished utilizing a cookie, or an explicit password entry, etc. Once the user is identified, a parameter file may be located in operation **420**. This may be accomplished automatically based on the identity of the user, or via a manual selection process. Next, the report is generated in operation **422** and stored for later viewing access in **424**.

Two different modes of report creation (**418**, **420**, **422** and **424**) are possible. Both are very useful in their specific environments:

An automatic or scheduled report creation mode may run, for example, every morning before the users come to the office. This is an easy way to create daily reports, based on previous data.

A manual mode may take dedicated dates to create the required reports. This may be a typical environment for Baselining projects, handled by some consultants. They save all network related data in a database. Later, they create the specific reports for their customers. Instead of creating every single report one by one, they can create the same report for 14 days on 10 segments (140 reports in total) with one simple command line.

Sniffer® Digest Homepage **324** in Fig. **3** can thus have a link to these reports or a link to an index, where all reports are referenced.

Figure **5** illustrates a method **500** how the WARE process **312** operates by using templates to create final documents, in accordance with operation **422** of Figure **4**. The WARE process **312** is started in **500** with some parameter file as input. The WARE process **312** reads all keywords in operation **502**. Keywords are kept in '[' ']' and start at the beginning of as line, so that the WARE process **312** can find them. Behind every keyword, a value is written. Table #1, which shows a sample WARE parameter file, illustrates this.

Operation **504** opens the template files based on some [INPUT] keyword. Parameters, which are identified by a ':' at the beginning and the end, are used to narrow down the data selection. Operation **506** scans the template file. Figure **5A** illustrates an exemplary template file. All placeholders in the template are put into '[', ']' **520**. Date placeholders are simply replaced in operation **508** with the date for which

the report should run. Standard date format options are possible. Other placeholders from the template file are also replaced with information from the parameter file.

Operation **510** takes a SQL statement, which is related to a placeholder and runs this against the database. During operation **512**, the results are replaced in the template. SQL statements like SQL_Q1 in Table #1 can be complex and may return several rows and columns. The related placeholders **522** in the template file also refer to the correct rows and columns to be printed. See again Figure **5A**. Another option, as shown by designator **524**, allows dynamic printing of the required records in the template. This is useful, if the returned row count is not known before the query and no useless space will be occupied in the final document. A specific '[[', ']]' syntax allows the WARE process **312** to automatically create new rows with the correct layout to print the data from the database. Only the minimum and maximum rows, which should be printed if they are available, are mentioned in the template.

Figure **5B** shows the result after all replacements from Figure **5A** are done. After all the replacements are done, the new document of Figure **5B** may be saved in a location, that is specified by the output keyword. See operation **514.** This operation can also use some placeholders in the output directory to be replaced with current parameters and dates.

The replacements in the template are done on visible and also invisible text like URLs. A general text replacement may be used. Therefore, also dynamic, result based links can be created automatically in the final report document. A complex set of flexible reports, linked to each other with easy user access and navigation can be the result of several WARE processes running one after the other. Different users can have different skeletons for their preferred reports and navigation.

Since the replacement may work as general text replacement, it is not only applicable for HTML files, but also for standard text, RTF and other text-based file formats.

Figure **6** illustrates a method **600** showing how the CARE process **316** operates by using some CrystalTM report templates. The CARE process **316** is started in **600** with some parameter file as input. The CARE process **316** reads all keywords in operation **602**. Keywords are kept in '[' ']' and start at the beginning of as line, so that CARE can find them. Behind every keyword a value is written. Table #2, which shows a sample CARE parameter file, illustrates this.

After the keyword PARAMETERS, the section that keeps all the parameters follows, which is used to feed into CrystalTM Reports. The keyword REPORT points to a CrystalTM report template, which was earlier designed with a CrystalTM report designer and now needs to be filled with current data. The SELECT keyword is used to control the data query, which CrystalTM will run against the database. This query can include several parameters, which are also found in the PARAMETERS section. Operation **604** creates the output directory, based on parameters, which are found in the PARAMTERS section of the parameter file.

All information is then sent in operation **606** through the CrystalTM API into the CrystalTM system for automated creation of the report. This involves the specific source location of the template, the destination location of the final report and any kind of information to customize the report. All of this is found in a parameter file as shown in Table #2. Different parameters can be used for looping and nesting. Table #3 shows the names of the directories, which may be automatically created based on the PARAMETERS of Table #2.

Table #3

```
D:\reports\2001_08_01\Dallas\WEBServer1\default.htm
D:\reports\2001_08_02\Dallas\WEBServer1\default.htm
D:\reports\2001_08_03\Dallas\WEBServer1\default.htm
D:\reports\2001_08_01\Dallas\SAPServer1\default.htm
D:\reports\2001_08_02\Dallas\SAPServer1\default.htm
D:\reports\2001_08_03\Dallas\SAPServer1\default.htm
D:\reports\2001_08_01\NewYork\WEBServer1\default.htm
D:\reports\2001_08_02\NewYork\WEBServer1\default.htm
D:\reports\2001_08_03\NewYork\WEBServer1\default.htm
D:\reports\2001_08_01\NewYork\SAPServer1\default.htm
D:\reports\2001_08_02\NewYork\SAPServer1\default.htm
D:\reports\2001_08_03\NewYork\SAPServer1\default.htm
D:\reports\2001_08_01\NewYork-Atlanta\WEBServer1\default.htm
D:\reports\2001_08_02\NewYork-Atlanta\WEBServer1\default.htm
D:\reports\2001_08_03\NewYork-Atlanta\WEBServer1\default.htm
D:\reports\2001_08_01\NewYork-Atlanta\SAPServer1\default.htm
D:\reports\2001_08_02\NewYork-Atlanta\SAPServer1\default.htm
D:\reports\2001_08_03\NewYork-Atlanta\SAPServer1\default.htm
D:\reports\2001_08_01\Dallas-Atlanta\WEBServer1\default.htm
D:\reports\2001_08_02\Dallas-Atlanta\WEBServer1\default.htm
D:\reports\2001_08_03\Dallas-Atlanta\WEBServer1\default.htm
D:\reports\2001_08_01\Dallas-Atlanta\SAPServer1\default.htm
D:\reports\2001_08_02\Dallas-Atlanta\SAPServer1\default.htm
D:\reports\2001_08_03\Dallas-Atlanta\SAPServer1\default.htm
```

Table #4 shows an example of a simple nested parameter file. The resulting directory structure is shown in Table #5.

Table #4

```
[REPORT]d:\reports\crystaltemplates\report.rpt

[EXPORT]d:\reports\:YEAR:_:MONTH:_:DAY:\:SEGMENT:\DLCI_:DLCI:\:H
OST:\default.htm
```

```
[SELECT] {HOSTS.Host} = ":HOST:" and
         {HOSTS.Time} in DateTime (:YEAR:,:MONTH:,:DAY:,00, 00,
00) to DateTime (:YEAR:,:MONTH:,:DAY:,23, 59, 59) and
         {HOSTS.Segment} = ":SEGMENT:" and
         {HOSTS.DLCI} = ":DLCI:"


[DATEFORMAT] %Y-%m-%d

[PARAMETERS]
:DATE: 2001-08-1
:DATE: 2001-08-2

:SEGMENT: Dallas
+:DLCI: 161
++:HOST: WEBServer1
++:HOST: WEBServer2
+:DLCI: 163
++ HOST: SAPServer1

:SEGMENT: NewYork
+:DLCI: 176
++:HOST: WEBServer1
+:DLCI: 177
++ HOST: SAPServer1
++ HOST: SAPServer2
```

Table #5

```
d:\reports\2001_08_01\Dallas\DLCI_161\WEBServer1\default.htm
d:\reports\2001_08_01\Dallas\DLCI_161\WEBServer2\default.htm
d:\reports\2001_08_01\Dallas\DLCI_163\SAPServer1\default.htm
d:\reports\2001_08_01\NewYork\DLCI_176\WEBServer1\default.htm
d:\reports\2001_08_01\NewYork\DLCI_177\SAPServer1\default.htm
d:\reports\2001_08_01\NewYork\DLCI_177\SAPServer2\default.htm
d:\reports\2001_08_02\Dallas\DLCI_161\WEBServer1\default.htm
d:\reports\2001_08_02\Dallas\DLCI_161\WEBServer2\default.htm
d:\reports\2001_08_02\Dallas\DLCI_163\SAPServer1\default.htm
d:\reports\2001_08_02\NewYork\DLCI_176\WEBServer1\default.htm
d:\reports\2001_08_02\NewYork\DLCI_177\SAPServer1\default.htm
d:\reports\2001_08_02\NewYork\DLCI_177\SAPServer2\default.htm
```

Instead of having only fixed values in the PARAMETER section, some SQL query can be used to create dynamic values, which are used for report creation. One sample parameter file is shown in Table #6.

Table #6

```
[REPORT]d:\reports\crystaltemplates\report.rpt

[EXPORT]d:\reports\:YEAR:_:MONTH:_:DAY:\:SEGMENT:\DLCI_:DLCI:\:H
OST:\default.htm


[SELECT] {HOSTS.Host} = ":HOST:" and
         {HOSTS.Time} in DateTime (:YEAR:,:MONTH:,:DAY:,00, 00,
00) to DateTime (:YEAR:,:MONTH:,:DAY:,23, 59, 59) and
         {HOSTS.Segment} = ":SEGMENT:"

[DATEFORMAT] %Y-%m-%d

[PARAMETERS]
:DATE: 2001-08-1
:DATE: 2001-08-2

:SEGMENT: Dallas
+:SQL,3: Select host from host_table where time >=
         CDate(':DATE: 00:00:00') and time <=
         CDate(':DATE: 23:59:59')
         group by hosts order by total_packets desc
+:SQLOUT: RECNR,HOST

:SEGMENT: NewYork
+:SQL,5: Select host from host_table where time >=
         CDate(':DATE: 00:00:00') and time <=
         CDate(':DATE: 23:59:59')
         group by hosts order by total_packets desc
+:SQLOUT: RECNR,HOST
```

This sample creates segment Dallas 3 reports (for the top 3 hosts) for 2 dates based on total packet counts. It also creates segment NewYork 5 reports (for the top 5 hosts) for 2 dates based on the total packet count on this segment. Since the network manager does not know the top hosts, the SQL query may make a dynamic decision, based on the current database.

Similar to the WARE process, the DATE information can either be absolute dates or relative dates, which are related to the date, when the job runs. The intention for scheduled or on-demand reports was already described in the description of Figure 5.

Operations **604** and **606** may loop until all reports, which are a result of the different parameter combinations, are created.

Figure **7** illustrates a snapshot of a possible output, which combines the CARE and WARE-process generated reports is shown in Figure **7**. As shown, an index **700** of the last days is provided. The creation of such can be done automatically by the WARE process. Designators **702, 704, 706** and **710** show values, which are a result from a query against the database. Item **704** is used to sort the entries by errors. Items **706** and **710** offer information about the utilization. Item **706** is a printed numeric value. As designated by designator **710**, a parameter for the bar size inside the html-script is replaced by the value of the utilization, which can only be a value somewhere between 0 and 100%. Item **708** is a date replacement, done by the WARE process. Item **702**, which is the segment name, also has a link in the background, which points to the location of the chart **712**. This link is also automatically created by the WARE process. The chart itself is the result of a CARE operation.

Different users can have different Sniffer® Digest HomePages, but they can all refer to a common pool of graphics, which is identified by dates, segment names and or host names. The required links are created automatically. There is no need to store all memory-consuming pictures several times. The CARE and WARE process can take care of the links and the locations.

Example

As an option, on the top left of the daily Sniffer® Digest Homepage, a graph may be shown, which prints the top five (5) error segments during the last day. Every day, the graph may be located at the same position, but the segments and their names may change based on every day's specific error occurrences.

Below this graph, a list of the top five (5) busiest servers may be shown. The list may be positioned at the same position every day, but the printed servers and their load may change. The top right of the page may show the same 'error graph' as on the top left. However, in this graph, the error count covers not only one day but always the last one week. Sniffer® Digest allows this way of customizing the reports in a very simple but effective way.

While various embodiments have been described above, it should be understood that they have been presented by way of example only, and not limitation. For example, any of the network elements may employ any of the desired functionality set forth hereinabove. Thus, the breadth and scope of a preferred embodiment should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.